

Security Considerations of a New Helper Data Scheme for Physical Unclonable Functions

Sven Muelich, Sven Puchinger, and Martin Bossert

Ulm University, Institute of Communications Engineering
`{sven.mueelich, sven.puchinger, martin.bossert}@uni-ulm.de`

Abstract. When Physical Unclonable Functions (PUFs) are used for cryptographic purposes, error correction in combination with a helper data scheme is an essential component due to the fuzzy nature of a PUF. All known schemes require both a code and additional helper data to recover PUF responses. Recently, Muelich and Bossert [1] proposed a scheme that only requires a code. In this paper, we answer two open questions about the new scheme. First, we show that under certain assumptions, the code construction within the scheme is possible with high probability. Second, it is proven that after constructing the code, the security level of the scheme is known and is sufficiently large with high probability.

1 Introduction

Physical Unclonable Functions (PUFs) exploit variations in the manufacturing process of hardware components in order to extract a unique and reproducible sequence of bits from each device. Due to their properties, these bit sequences can be used for example as cryptographic keys. Since the bit sequences are reproducible, they can be regenerated on demand and do not have to be stored in a non-volatile memory. Hence, some kinds of physical attacks that aim for reading the key from the memory are avoided.

However, when re-extracting a key, errors occur because of physical reasons like measurement noise or environmental conditions. For the error correction process, so-called *Helper Data Algorithms* are used. Helper data are extracted from an initial response and stored in a non-volatile memory. There is no need to use a protected memory, since the algorithms are designed such that an attacker with access to the helper data does not have less uncertainty about the key than without knowing the helper data. The helper data are used for a pre-processing stage within the error correction phase during key reproduction.

Recently, [1] suggested the first helper data algorithm which only uses an error correcting code without any additional helper data. This work investigates practicability and security issues of the new scheme. In Section 3, we present an analysis of the probability that the construction of a code of given length and dimension is possible with this scheme. The analysis is based on assumptions on the underlying LDPC codes and verified using practical experiments.

In Section 4, we present a lower bound on the uncertainty of an attacker about the input of the code construction algorithm that only depends on the rank of a matrix which is known after constructing the code. It is then shown that in case of a successful code construction, the uncertainty is always sufficiently large.

2 Preliminaries

2.1 Notation

Let $\mathcal{V} \subseteq \mathbb{F}_2^n$ be a subspace. Then,

$$\text{OC}(\mathcal{V}) := \{\mathbf{u} \in \mathbb{F}_2^n : \mathbf{u}\mathbf{v}^T = 0 \forall \mathbf{v} \in \mathcal{V}\}$$

is the *orthogonal complement* of \mathcal{V} . We know that $\dim(\text{OC}(\mathcal{V})) = n - \dim(\mathcal{V})$. Let \mathbf{H} be a matrix. Then, the *row space* of \mathbf{H} is denoted by $\langle \mathbf{H} \rangle$. For convenience, we write $\text{OC}(\mathbf{H}) := \text{OC}(\langle \mathbf{H} \rangle)$. Note that in coding theory, the orthogonal complement of a code is often called the *dual code*.

A *parity check matrix* of a linear code \mathcal{C} is a basis of the orthogonal complement $\text{OC}(\mathcal{C})$. Any other matrix \mathbf{H} whose rows span $\text{OC}(\mathcal{C})$ is called a *decoding matrix*.

2.2 Physical Unclonable Functions (PUFs)

Physical Unclonable Functions (PUFs) can be used for cryptographic purposes like identification, authentication and secure key generation. A PUF extracts a unique and reproducible sequence of bits, which is called response, from an integrated circuit. The uniqueness results from randomness which is intrinsic to each device, due to technical and physical variations within the manufacturing process. Since these variations cannot be controlled by the manufacturer, devices which exploit this behavior are called physically unclonable, since the behavior cannot be cloned. Usually either the delay behavior of a device (e.g. Ring-Oscillator PUFs [2]) or the initialization behavior of memory cells (e.g. SRAM PUFs [3]) is used in order to extract responses. An extracted response can for example be used as cryptographic key due to its uniqueness.

The advantage of using PUFs for key generation is, that there is no need to store the key in a non-volatile memory, since it can be simply reproduced when it is needed by the cryptosystem. Storing a key somewhere in a non-volatile memory makes a system vulnerable to physical attacks, even when a protected memory is used [4]. Unfortunately, due to measurement noise or environmental conditions like temperature, supply voltage or aging of the chip, errors occur when the response is repeatedly extracted. For this reason, error correction is needed. For details regarding different PUF constructions we refer to the literature [5–7].

2.3 Helper Data Algorithms

Different Helper Data algorithms have been proposed in the literature. The most popular ones are *Code-Offset Construction* and *Syndrome Construction* [8, 9]. Both have in common, that an error correcting code as well as additional helper data are required in order to reproduce a PUF response. Helper data algorithms consist of two phases, *initialization phase* and *reproduction phase*. In the initialization phase, helper data are extracted from an initial PUF response. This phase is performed only once within a secure environment during the manufacturing process of the device. In the reproduction phase, these helper data are used in order to reproduce the initial response from a regenerated, erroneous response. The reproduction phase occurs in field whenever the cryptosystem needs access to the key.

In the initialization phase of the Code-Offset Construction, a codeword \mathbf{c} from a specified code \mathcal{C} is randomly chosen and added bitwise to the initial response \mathbf{r}_I . The model assumes that $\mathbf{r}_I \sim \mathcal{U}(\mathbb{F}_2^n)$. The bit sequence \mathbf{h} which is obtained by this operation is stored as helper data in a non-volatile, possibly non-protected helper data storage. In the reproduction phase, which is executed in the field as often as the cryptosystem needs the key, the helper data \mathbf{h} are added to a new extracted response \mathbf{r} in a pre-processing step of the error correction. The result of this operation has the form $\mathbf{c} + \mathbf{e}$ where \mathbf{e} is a vector of low weight and hence can be interpreted as error vector. A decoder for the chosen code can be used in order to calculate \mathbf{c} . Adding again the helper data \mathbf{h} to \mathbf{c} yields the initial response \mathbf{r}_I which was extracted in the initialization phase.

Recently, [1] proposed a scheme which only uses a code without additional helper data. The main idea of this new scheme is to construct a code \mathcal{C} , such that the initial PUF response is a codeword of \mathcal{C} . This construction happens in the initialization phase of the scheme and is discussed in detail in Section 2.4. The response \mathbf{r} , which is extracted in the reproduction phase, slightly differs from the initial response \mathbf{r}_I and hence can be described as $\mathbf{r} = \mathbf{r}_I + \mathbf{e}$, where \mathbf{e} is again a vector of low weight. If a decoder for the constructed code \mathcal{C} is used, \mathbf{r}_I can be recovered by simply decoding \mathbf{r} (cf. Algorithm 1). The essential difference between this new scheme and previous schemes is that only a code is needed, but no further helper data are required. This directly implicates, that there is no need for an additional pre-processing step before decoding can take place. Figure 1 visualizes the initialization and reproduction phase of the new scheme.

Algorithm 1: Helper Data Algorithm: Reproduction phase [1]

Input: Re-extracted PUF response $\mathbf{r} = \mathbf{r}_I + \mathbf{e} \in \{0, 1\}^n$, \mathbf{H}

Output: Decoding result $\hat{\mathbf{r}} = \text{dec}(\mathbf{r}, \mathbf{H})$

1 $\hat{\mathbf{r}} = \text{dec}(\mathbf{r}, \mathbf{H})$

2 **return** $\hat{\mathbf{r}}$

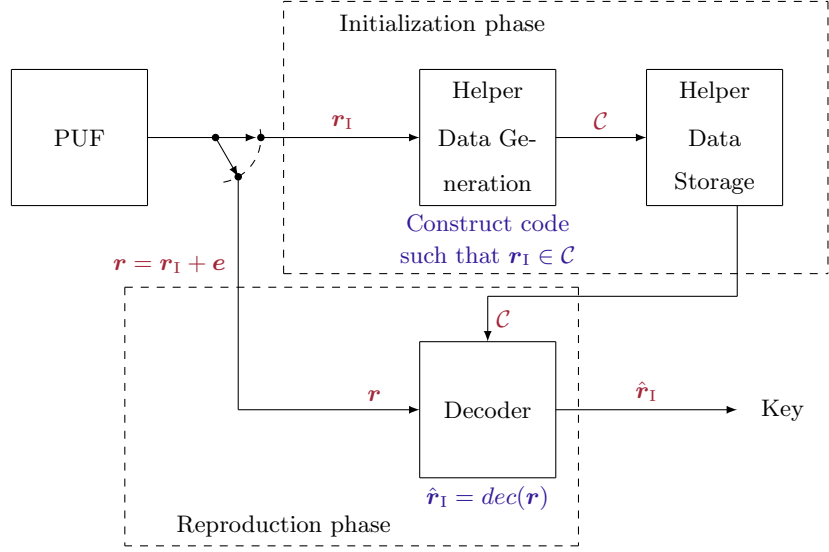


Fig. 1. Initialization and reproduction phase of the helper data algorithm [1].

2.4 Code Construction

Let $\mathbf{H}^{(I)}$ be an $m_{\mathbf{H}^{(I)}} \times n$ matrix of rank $rk_{\mathbf{H}}$ that is sparse, where $m_{\mathbf{H}^{(I)}}$ is allowed to be larger than $rk_{\mathbf{H}^{(I)}}$. The matrix can be interpreted as a decoding matrix of an LDPC code of length n and dimension $n - rk_{\mathbf{H}^{(I)}}$. In [1], the rows of $\mathbf{H}^{(I)}$ were chosen as the union of the rows of different decoding matrices $\mathbf{H}_i^{(I)}$ of various constructions of LDPC codes \mathcal{C}_i of length n (Euclidean Geometry [10], Projective Geometry [10], and Reed–Solomon based [11] codes). Table 1 shows for some examples how much larger $m_{\mathbf{H}^{(I)}}$ is compared to $rk_{\mathbf{H}^{(I)}}$.

n	k	$m_{\mathbf{H}^{(I)}}$	$rk_{\mathbf{H}^{(I)}}$
128	13	881	115
128	56	349	72
256	106	555	150

Table 1. Code examples [1].

The aim is to find an LDPC code that contains the initial response r_I . Algorithm 2 describes the method from [1] for constructing a decoding matrix \mathbf{H} from r_I and $\mathbf{H}^{(I)}$.

Algorithm 2: Helper Data Algorithm: Initialization phase [1]

Input: PUF response $\mathbf{r} \in \{0, 1\}^n$, decoding matrix $\mathbf{H}^{(1)}$ of an LDPC code with $m_{\mathbf{H}^{(1)}}$ rows

Output: Decoding matrix \mathbf{H} of a code $\mathcal{C}(n, k)$, such that $\mathbf{r} \in \mathcal{C}$

```
1  $\mathbf{H}$  is matrix with  $n$  columns and 0 rows.  
2 for  $i = 1, 2, \dots, m_{\mathbf{H}^{(1)}}$  do  
3   if  $i$ -th row  $\mathbf{h}_i$  of  $\mathbf{H}^{(1)}$  is orthogonal to  $\mathbf{r}_1$  then  
4      $\perp$  Vertically append  $\mathbf{h}_i$  to  $\mathbf{H}$ .  
5 return  $\mathbf{H}$ 
```

Since an attacker knows the code and we would like the attacker's uncertainty to be beyond a given *security level* SL (e.g., $SL \approx 128$), we require that the dimension k of the resulting code is greater or equal the security level SL . This means that \mathbf{H} should have at most rank $n - SL$. On the other hand, the code should have a good error-correction performance and hence, the rank should also not be chosen too small.

In this paper, we answer two questions that have not been addressed in [1]:

- i) Is such a construction possible for all initial responses \mathbf{r}_1 ?
- ii) Is it possible to reduce the attacker's uncertainty below k by considering the deterministic behavior of Algorithm 2?

3 Successful Code Constructions

In this section, we would like to analyze for which responses \mathbf{r}_1 it is possible to construct an LDPC code \mathcal{C} of sufficiently small dimension k that contains \mathbf{r}_1 as codeword. For instance, if k should be at most k_{\max} , we need the output matrix \mathbf{H} of Algorithm 2 to have at least rank $n - k_{\max}$. We start by analyzing the problem theoretically under certain assumptions and verify the results using practical experiments.

3.1 Theoretical Ideas

Let \mathbf{h}_i be the i -th row of $\mathbf{H}^{(1)}$. The probability of \mathbf{h}_i being orthogonal to \mathbf{r}_1 depends on the number of ones in the positions of \mathbf{r}_1 which are indexed by the support of \mathbf{h}_i . Since $\mathbf{r}_1 \sim \mathcal{U}(\mathbb{F}_2^n)$, it follows that

$$\Pr(\mathbf{r}_1 \cdot \mathbf{h}_i^T = 0) = \frac{1}{2}.$$

Since \mathbf{H} is sparse, the support of two distinct rows \mathbf{h}_i and \mathbf{h}_j of $\mathbf{H}^{(1)}$ is unlikely to overlap, and in case of an overlap, it only concerns a small number of positions. Therefore, we assume that the events $\mathbf{r}_1 \cdot \mathbf{h}_i^T = 0$ are statistically independent.

Thus, the number of rows $m_{\mathbf{H}}$ of \mathbf{H} is binomially distributed with parameters $m_{\mathbf{H}^{(1)}}$ and $\frac{1}{2}$, i.e.,

$$m_{\mathbf{H}} \sim \text{Bin}(m_{\mathbf{H}^{(1)}}, \frac{1}{2}).$$

We will show in Section 3.2 that this can be verified for practical examples.

We can thus conclude that if the decoding matrix $\mathbf{H}^{(1)}$ contains many more rows than its rank, i.e., $m_{\mathbf{H}^{(1)}} \gg rk_{\mathbf{H}^{(1)}}$, then the probability of having less than $rk_{\mathbf{H}^{(1)}}$ rows in \mathbf{H} is negligible. Also, the rank of \mathbf{H} fulfills

$$\text{rank}(\mathbf{H}) \geq \text{rank}(\mathbf{H}^{(1)}) - 1$$

with large probability since the $m_{\mathbf{H}} \gg rk_{\mathbf{H}^{(1)}}$ rows of \mathbf{H} are very likely to contain a generating set of

$$\mathcal{V} := \langle \mathbf{H}^{(1)} \rangle \cup \text{OC}(\mathbf{r}_I)$$

and

$$\begin{aligned} \dim(\mathcal{V}) &= \dim(\langle \mathbf{H}^{(1)} \rangle) + \dim(\text{OC}(\mathbf{r}_I)) - \underbrace{\dim(\langle \mathbf{H}^{(1)} \rangle + \text{OC}(\mathbf{r}_I))}_{\leq n} \\ &\geq \text{rank}(\mathbf{H}^{(1)}) - 1. \end{aligned}$$

3.2 Numerical Results

We verify the results in Section 3.1 in a practical scenario. The example uses an (512, 139) Euclidean Geometry code, whose decoding matrix $\mathbf{H}^{(1)}$ has $m_{\mathbf{H}^{(1)}} = 4672$ rows. We generated random responses $\mathbf{r}_I \sim \mathcal{U}(\mathbb{F}_2^n)$ and used Algorithm 2 to obtain an LDPC code with decoding matrix \mathbf{H} containing \mathbf{r}_I . Using the results of Section 3.1, the number of rows $m_{\mathbf{H}}$ should be approximately $\text{Bin}(m_{\mathbf{H}^{(1)}} = 4672, 0.5)$ distributed. Figure 2 visualizes the empirical cdf (sample size = 10^6) in comparison to the theoretical cdf.

It can be seen that the two curves almost coincide. The estimated mean and variance of $m_{\mathbf{H}}$ are $\hat{\mu} \approx 2335.99$ and $\hat{\sigma}^2 \approx 1169.49$. This is close to the theoretical values $E(m_{\mathbf{H}}) = \frac{m_{\mathbf{H}^{(1)}}}{2} = 2336$ and $\text{Var}(m_{\mathbf{H}}) = \frac{m_{\mathbf{H}^{(1)}}}{4} = 1168$. In addition, it was observed that $\text{rank}(\mathbf{H}) = 272 = \text{rank}(\mathbf{H}^{(1)}) - 1$ for all tested cases (sample size = 10^3 due to larger complexity of rank computation), which also coincides with the theoretical prediction.

4 Security Issues

Since the code \mathcal{C} contains 2^k codewords, including \mathbf{r}_I , the uncertainty of \mathbf{r}_I for an attacker is at most k . In this section, we derive a lower bound on the uncertainty and show that in most cases, it is lower-bounded by $k - 2$.

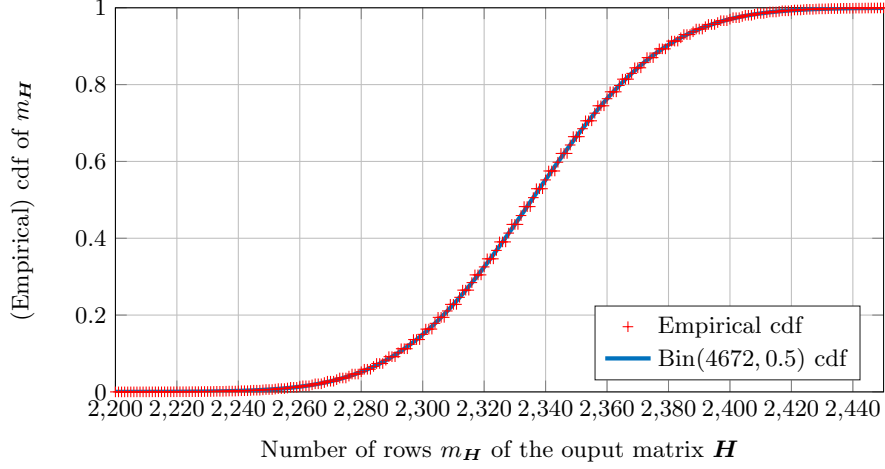


Fig. 2. Comparison of the empirical cdf of the number of rows m_H of H with the cdf of a $\text{Bin}(m_{H^{(1)}}, 0.5)$ binomially distributed random variable. Sample size 10^6 , number of rows $m_{H^{(1)}} = 4672$ of $H^{(1)}$.

4.1 Uncertainty of an Attacker

Let $H^{(1)}$ denote the original decoding matrix and \widetilde{H} be the rows of $H^{(1)}$ that are orthogonal to r_I and \overline{H} be the other rows. The “attack” described above, which uses the fact that r_I is a codeword of \mathcal{C} , or in other words

$$r_I \in \mathcal{C} := \text{OC}(\widetilde{H}), \quad (1)$$

is suboptimal since it does not make use of \overline{H} . We can in addition use the latter by the following observation:

$$r_I \in \bigcap_{i=1}^{m_{\overline{H}}} \mathcal{V}_i, \quad (2)$$

where $\mathcal{V}_i := \mathbb{F}_2^n \setminus \text{OC}(h_i)$ with the i -th row h_i of \overline{H} . By combining Equations (1) and (2), we obtain

$$r_I \in \mathcal{C} \cap \left(\bigcap_{i=1}^{m_{\overline{H}}} \mathcal{V}_i \right) =: \mathcal{S}.$$

The uncertainty of an attacker $H(r_I)$ about r_I is directly determined by the size of \mathcal{S} . More precisely, since $r_I \in \mathcal{U}(\mathbb{F}_2^n)$, the uncertainty is lower-bounded by

$$H(r_I) \geq \log_2(|\mathcal{S}|).$$

We use the following theorem to derive a lower bound on $H(r_I)$.

Theorem 1. A response \mathbf{r}_1 fulfills Conditions (1) and (2) if and only if

$$\mathbf{r}'_1 \in \text{OC}(\mathbf{H}'),$$

where

$$\begin{aligned} \mathbf{r}'_1 &:= [\mathbf{r}_1, 1] \in \mathbb{F}_2^{n+1}, \\ \mathbf{H}' &:= \begin{bmatrix} \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \\ \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \end{bmatrix} \in \mathbb{F}_2^{(m_{\widetilde{\mathbf{H}}} + m_{\overline{\mathbf{H}}}) \times (n+1)}. \end{aligned}$$

Proof. Condition (1) is fulfilled if and only if

$$\mathbf{h}_i \cdot \mathbf{r}'_1 = 0 \quad \forall i,$$

where \mathbf{h}_i is the i -th row of $\widetilde{\mathbf{H}}$. This again holds if and only if

$$[\mathbf{h}_i, 0] \cdot [\mathbf{r}_1, 1]^T = 0 \quad \forall i,$$

where $[\mathbf{h}_i, 0]$ is the i -th row of $\begin{bmatrix} \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \\ \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \end{bmatrix}$, which means that $[\mathbf{r}_1, 1]$ is in the right kernel of $\begin{bmatrix} \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \\ \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \end{bmatrix}$. Similarly, Condition (2) is satisfied if and only if

$$\mathbf{h}_i \cdot \mathbf{r}'_1 = 1 \quad \forall i,$$

where \mathbf{h}_i is the i -th row of $\overline{\mathbf{H}}$. This implies that

$$[\mathbf{h}_i, 1] \cdot [\mathbf{r}_1, 1]^T = 0 \quad \forall i,$$

where $[\mathbf{h}_i, 1]$ is the i -th row of $\begin{bmatrix} \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \\ \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \end{bmatrix}$, which means that $[\mathbf{r}_1, 1]$ is in the right kernel of $\begin{bmatrix} \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \\ \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \end{bmatrix}$. Since the rows of $\begin{bmatrix} \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \\ \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \end{bmatrix}$ and $\begin{bmatrix} \overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1} \\ \widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1} \end{bmatrix}$ are exactly those of \mathbf{H}' , we get that the conditions are fulfilled if and only if $[\mathbf{r}_1, 1]$ is in the right kernel of \mathbf{H}' . \square

Using the statement of Theorem 1, the cardinality of $\log_2(\mathcal{S})$, and therefore a lower bound on the attacker's uncertainty about \mathbf{r}_1 , is directly determined by the rank of the matrix \mathbf{H}' in the following way.

Corollary 1. $H(\mathbf{r}_1) \geq \log_2(|\mathcal{S}|) = n - \text{rank}(\mathbf{H}')$.

Proof. Using the rank nullity theorem (*), we get

$$\log_2(\mathcal{S}) = \dim(\text{OC}(\mathbf{H}')) - 1 \stackrel{(*)}{=} (n+1) - \text{rank}(\mathbf{H}') - 1 = n - \text{rank}(\mathbf{H}'),$$

which proves the claim. \square

Since the matrix \mathbf{H}' is directly known after constructing the code \mathcal{C} , Corollary 1 provides a tool to determine a lower bound on the uncertainty of the attacker. If the uncertainty is too low, the PUF can be removed from the set of suitable PUFs. This is no problem since it is done during the manufacturing process. We will see in the next subsection that $\log_2(|\mathcal{S}|)$ is at least $k - 2$ with large probability, where k is the dimension of \mathcal{C} .

4.2 Practical Considerations

In Section 4.1, we have derived a lower bound on an attacker's uncertainty about \mathbf{r}_I which only depends on the rank of the matrix \mathbf{H}' . The following theorem shows that this rank is sufficiently small for the case of a successful code construction, i.e., $\text{rank}(\widetilde{\mathbf{H}}) \geq \text{rank}(\mathbf{H}^{(I)}) - 1$.

Theorem 2. *If $\text{rank}(\widetilde{\mathbf{H}}) \geq \text{rank}(\mathbf{H}^{(I)}) - 1$ then $\text{rank}(\mathbf{H}') \leq \text{rank}(\widetilde{\mathbf{H}}) + 2$.*

Proof. Due to $\text{rank}(\widetilde{\mathbf{H}}) \geq \text{rank}(\mathbf{H}^{(I)}) - 1$, we get

$$\begin{aligned} \dim(\langle \widetilde{\mathbf{H}} \rangle \cap \langle \overline{\mathbf{H}} \rangle) &= \underbrace{\text{rank}(\widetilde{\mathbf{H}})}_{\geq \text{rank}(\mathbf{H}^{(I)}) - 1} + \text{rank}(\overline{\mathbf{H}}) - \underbrace{\dim(\langle \widetilde{\mathbf{H}} \rangle + \langle \overline{\mathbf{H}} \rangle)}_{= \text{rank}(\mathbf{H}^{(I)})} \\ &\geq \text{rank}(\overline{\mathbf{H}}) - 1. \end{aligned}$$

Hence, since $\langle \widetilde{\mathbf{H}} \rangle \cap \langle \overline{\mathbf{H}} \rangle$ is a subspace of $\langle \overline{\mathbf{H}} \rangle$ of dimension at least $\text{rank}(\overline{\mathbf{H}}) - 1$, the vector space $\langle \overline{\mathbf{H}} \rangle$ is a direct sum of the form

$$\langle \overline{\mathbf{H}} \rangle = \langle \mathbf{h} \rangle + (\langle \widetilde{\mathbf{H}} \rangle \cap \langle \overline{\mathbf{H}} \rangle),$$

where $\mathbf{h} \in \langle \overline{\mathbf{H}} \rangle$. This means that all rows of the lower half of \mathbf{H}' , i.e., $[\overline{\mathbf{H}} \mathbf{1}_{m_{\overline{\mathbf{H}}} \times 1}]$, are of the form

- (i) $[\mathbf{h} + \tilde{\mathbf{h}}, 1]$, where $\tilde{\mathbf{h}}$ is in the span of the rows of $\widetilde{\mathbf{H}}$.
- (ii) $[\tilde{\mathbf{h}}, 1]$, where $\tilde{\mathbf{h}}$ is in the span of the rows of $\widetilde{\mathbf{H}}$.

If there is at least one row $\mathbf{h}'_1 = [\mathbf{h} + \tilde{\mathbf{h}}', 1]$ of type (i), then all such rows are in the span of $[\widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1}]$ and \mathbf{h}'_1 since

$$[\mathbf{h} + \tilde{\mathbf{h}}, 1] = \underbrace{[\mathbf{h} + \tilde{\mathbf{h}}', 1]}_{= \mathbf{h}'_1} + \underbrace{[\tilde{\mathbf{h}} - \tilde{\mathbf{h}}', 0]}_{\in \langle [\widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1}] \rangle}.$$

A similar argument holds for rows of type (ii), if there is one such row \mathbf{h}'_2 .

Hence, the rows of \mathbf{H}' are spanned by the rows of the matrix $\langle [\widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1}] \rangle$ and \mathbf{h}'_1 and \mathbf{h}'_2 and its rank is

$$\text{rank}(\mathbf{H}') \leq \text{rank}([\widetilde{\mathbf{H}} \mathbf{0}_{m_{\widetilde{\mathbf{H}}} \times 1}]) + 2 = \text{rank}(\widetilde{\mathbf{H}}) + 2,$$

which proves the claim. \square

Thus, in the case of a successful code construction, by Corollary 1, the uncertainty of the attacker about the response \mathbf{r}_I is lower-bounded as follows.

Corollary 2. *If $\text{rank}(\widetilde{\mathbf{H}}) \geq \text{rank}(\mathbf{H}^{(I)}) - 1$, then $H(\mathbf{r}_I) \geq k - 2$.*

Recall that we have shown in Section 3 that the assumption $\text{rank}(\widetilde{\mathbf{H}}) \geq \text{rank}(\mathbf{H}^{(I)}) - 1$ is fulfilled with high probability and therefore, with high probability, we get that $H(\mathbf{r}_I) \geq k - 2$.

5 Conclusion

In this work, we have shown the practicability of the helper data scheme for error correction in PUFs which was introduced in [1]. More precisely, the code construction, which is part of the scheme, is possible with high probability for desired code length and dimension. Also, we have derived a lower bound on the uncertainty about the PUF response of an attacker. Using this bound, we have obtained a tool to check the security level (SL) of the scheme given the constructed code. In most cases, the scheme is sufficiently secure when the dimension k of the code is chosen as $k \geq SL + 2$.

References

1. S. Muelich and M. Bossert, “A New Error Correction Scheme for Physical Unclonable Functions,” in *ITG SCC*, 2017.
2. G. E. Suh and S. Devadas, “Physical Unclonable Functions for Device Authentication and Secret Key Generation,” in *Proceedings of 44th annual Design Automation Conference*. ACM, 2007, pp. 9–14.
3. J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “FPGA Intrinsic PUFs and their Use for IP Protection,” in *CHES*. Springer, 2007, pp. 63–80.
4. R. Torrance and D. James, “The State-of-the-art in IC Reverse Engineering,” in *CHES*. Springer, 2009, pp. 363–381.
5. C. Boehm and M. Hofer, *Physical Unclonable Functions in Theory and Practice*. Springer, 2013.
6. R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer, 2013.
7. C. Wachsmann and A. Sadeghi, *Physically Unclonable Functions (PUFs): Applications, Models, and Future Directions*. Morgan & Claypool, 2015.
8. Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and other Noisy Data,” in *Advances in Cryptology-Eurocrypt*. Springer, 2004, pp. 523–540.
9. J.-P. Linnartz and P. Tuyls, “New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates,” in *Audio- and Video-based Biometric Person Auth.* Springer, 2003, pp. 393–402.
10. Y. Kou, S. Lin, and M. P. Fossorier, “Low Density Parity Check Codes: Construction Based on Finite Geometries,” in *IEEE Global Telecommunications Conference*, vol. 2, 2000, pp. 825–829.
11. I. Djurdjevic, J. Xu, K. Abdel-Ghaffar, and S. Lin, “A Class of Low-Density Parity-Check Codes Constructed based on Reed-Solomon Codes with two Information Symbols,” in *Int. Symp. on Appl. Algebra, Algebraic Algorithms, and Error-Correcting Codes*. Springer, 2003, pp. 98–107.